Informatica Olympiade 2021-2022

Handreiking en tips van Wt
Stedelijk Gymnasium 's-Hertogenbosch

Wat is de Informatica Olympiade?

De Nederlandse Informatica Olympiade (NIO) is een programmeerwedstrijd voor de bovenbouw van het Voortgezet onderwijs. Het is een onderdeel van de International Olympiad in Informatics (IOI).

Hij werkt wat anders dan de olympiades van de andere vakken. Je kunt in je eigen tijd en tempo aan de opgaven werken en zo vaak nieuwe versies van je uitwerking maken als je wilt. De hoogste score bij een vraag blijft staan. Als de deadline verstrijkt (half januari) dan kun je niets meer insturen en wordt je score definitief.

De eerste ronde van de NIO bestaat uit 4 onderdelen A, B, C en D:

Soort	Omschrijving	Aantal	Punten per opgave	Totaal te behalen
A	Inleidende opgaven	5	40	200
B	Theoretische opgaven	4	50	200
C	Gevorderde opgaven	2	100	200
D	Een spel programmeren	1	100	100

Opgaven en links:

- **PDF met alle opgaven van de NIO 2022**
- Digitaal inleverplatform van de NIO
- Hoofdwebsite van de NIO

De 150 inzendingen met de meeste punten (mits deze minimaal 200 punten hebben gehaald) worden uitgenodigd voor de tweede ronde van de NIO in maart 2022 (vorig jaar lag de grens op 256 punten, dus vermoedelijk zal het dit jaar vergelijkbaar zijn). Normaliter is de 2^{e} ronde op de Technische Universiteit Twente, maar is afhankelijk van de corona ontwikkeleingen.

Om mee te doen aan de olympiade, moeten je uitwerkingen digitaal zijn ingeleverd bij de NIO voor 16 januari 2022. (Ze worden bij inleveren direct automatisch nagekeken en je kunt zo vaak nieuwe/verbeterde versies insturen als je wilt). De beste deelnemers van de tweede ronde gaan door naar de derde ronde om te bepalen wie er namens Nederland mee mag doen aan de Internationale Informatica Olympiade in de zomer van 2021 in Singapore.

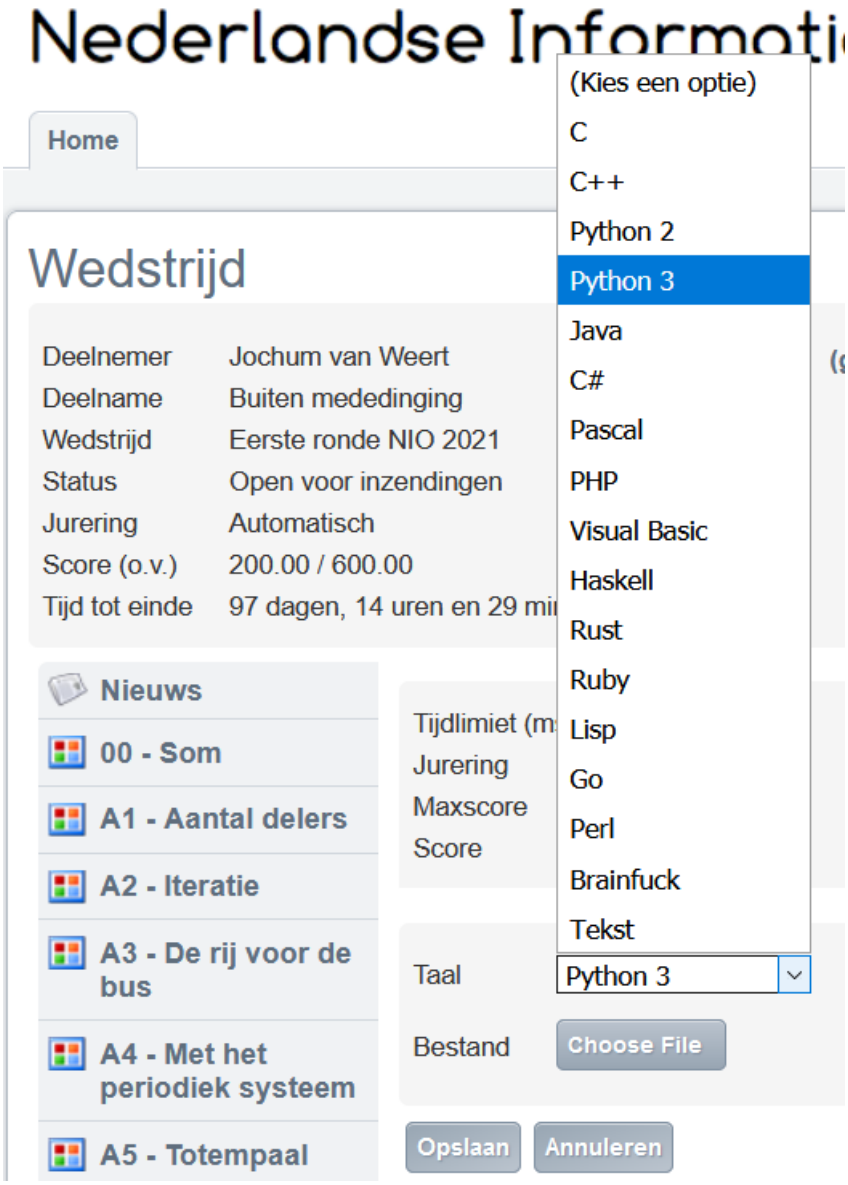
Automatisch nakijken

De informatica olympiade werkt met een automatisch nakijksysteem. Hierbij kun je een script dat je geschreven hebt als oplossing voor een van de opgaven inleveren. Er worden dan een aantal testcases automatisch op losgelaten en je krijgt punten voor elke testcase waarvoor je script het juiste antwoord geeft. Voor de testcases die misgaan krijg je te zien wat de foutmelding was (bv. Een fout antwoord, een error bij het runnen, maximale runtime overschreden, etc.). Met deze info kun je je script eventueel verbeteren en weer een nieuwe versie indienen. Je kunt elke opgave zo vaak proberen als je wilt, de hoogste score blijft staan. Het werkt als volgt:

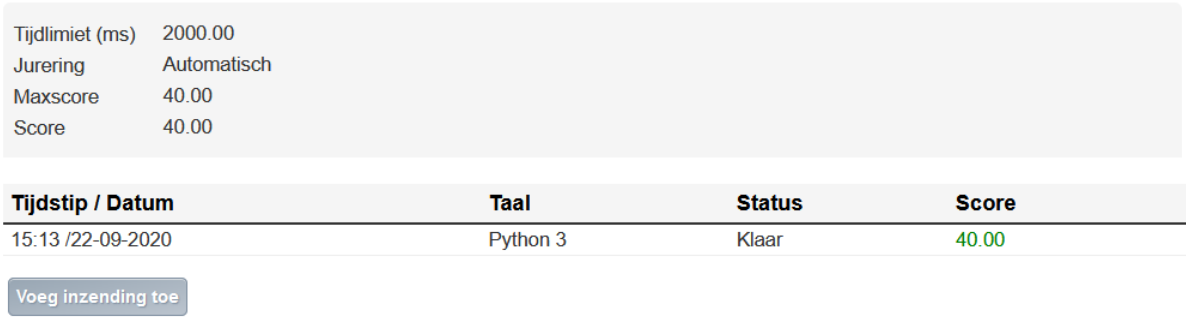
Ga naar http://submit.informaticaolympiade.nl/ en maak een account aan (geef Wt even als docent aan).

Als je bent ingelogd zie je links alle A, B en C opgaven staan (en 00 is een testopgave voor als je wilt checken of het inleveren werkt). Klik op de opgave je je wilt inleveren en klik “Voeg inzending toe”.

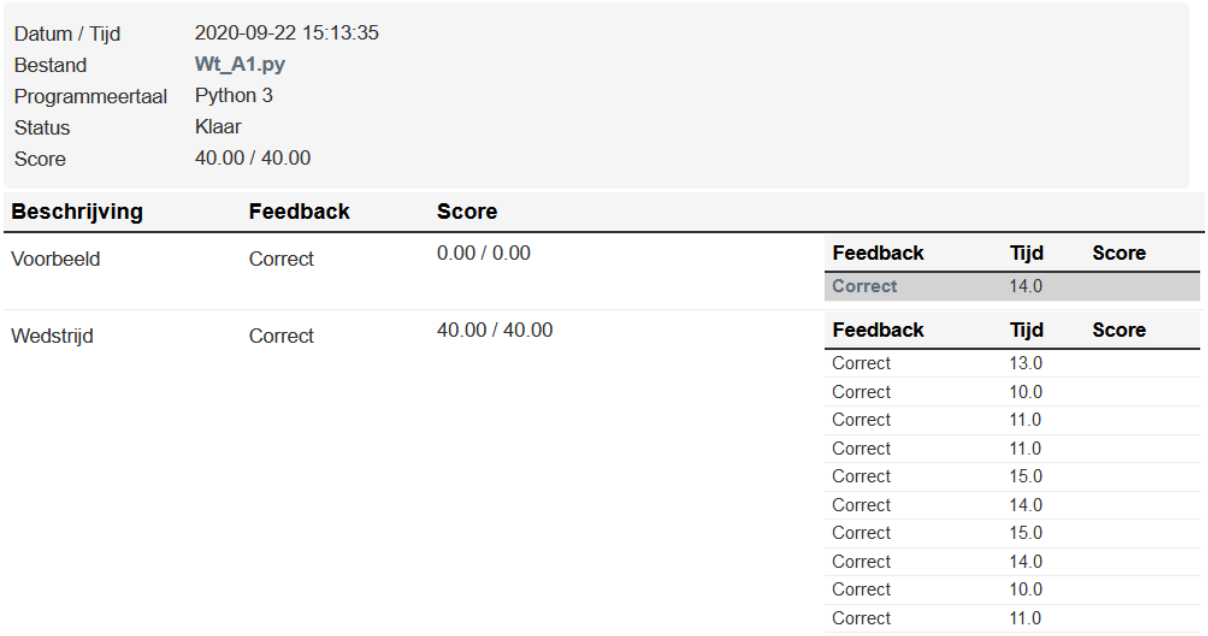
Je kunt nu een script uploaden. Vergeet niet de juiste programmeertaal te selecteren (ik kies hier Python 3, maar als je een andere favoriete taal hebt kan dat ook gewoon):



Als je op opslaan klikt wordt je script geüpload en uitgevoerd. Je ziet dan je inzending in de lijst staan met de status “uitvoeren”. Dit betekent dat je script op dit moment uitgevoerd wordt. Als je na een paar seconden de pagina ververst (F5), dan zie je als het goed is de status op “klaar” staan en is er een score gegeven (max 40 voor elke A opgave):



Je kunt nu op de inzending klikken voor meer info en eventuele foutmeldingen als je niet alle 40 punten hebt gehaald:



Datum / Tijd	2020-09-22 15:13:35
Bestand	Wt_A1.py
Programmeertaal	Python 3
Status	Klaar
Score	40.00 / 40.00

Beschrijving	Feedback	Score
Voorbeeld	Correct	0.00 / 0.00
Wedstrijd	Correct	40.00 / 40.00

Feedback	Tijd	Score
Correct	14.0	

Feedback	Tijd	Score
Correct	13.0	
Correct	10.0	
Correct	11.0	
Correct	11.0	
Correct	15.0	
Correct	14.0	
Correct	15.0	
Correct	14.0	
Correct	10.0	
Correct	11.0	

Als er foutmeldingen staan in de rechtertabel, kunnen die je helpen de fouten in je script op te sporen.

Algemene tips bij de A opgaven (voor python)

- Bij elk van deze opgaven wordt er 1 of meer keer om input gevraagd (bij het automatisch nakijken wordt je programma getest met verschillende inputs). Gebruik hiervoor steeds simpelweg `input()`
  De Olympiadesoftware gaat ervan uit dat je precies input leest zoals in de opgave is opgegeven en dat je precies het antwoord geeft zoals is voorgeschreven. Je moet dus geen extra tekst of tussenuitkomsten afdrukken. Doe dus ook niet zoiets als `input("Geef een getal")`, maar simpelweg `input()` (met niets tussen de haakjes), zodat er niets extra's op het scherm wordt geprint. Je opgave wordt anders niet goed gerekend door de nakijksoftware.
- Om je programma's te testen is het vervelend steeds de invoer te moeten intypen. Zet tijdens het testen de waardes waar je mee wilt testen rechtstreeks in de code (en commentaar de "input" even weg). Dat scheelt je een hoop typewerk.
  Niet vergeten de "input" weer terug te zetten voordat je de opgave instuurt naar de nakijksoftware!
- Voor het oplossen van opgave 4 (en vermoedelijk nog meer daarna) heb je een programmeerprincipe nodig dat **recursie** heet. Recursie is programmeren met een functie (`def`) die in de code van de functie zichzelf aanroept (en daarbij dus zichzelf weer aanroept, die daarbij zichzelf weer aanroept, etc.) Voor een goede introductie op recursie, kijk eens hier: http://programarcadegames.com/index.php?chapter=recursion&lang=nl#section_19
  (Deze link werkt niet op school vanwege de firewall, Een werkende Wt kopie staat hier)

Een bekend voorbeeld van recursie is het berekenen van de Fibonacci reeks Hier is een recursieve oplossing voor het Fibonacci probleem:

```
def fib(n):
    if n==1 or n==2:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

Bekijk (en test) het voorbeeld eens om goed te snappen wat er gebeurt. Je ziet dat de functie zichzelf steeds 2x aanroept (dat is de “recursiestap”) om zo de vorige 2 Fibonacci getallen te berekenen die hij nodig heeft om de huidige te bepalen. Omdat fib(1) en fib(2) wel gewoon constant zijn (dit noem je de “basis-stap” van de recursie), eindigt de recursie uiteindelijk en wordt er een antwoord gegenereerd.

Vergelijk de recursieve oplossing met de gewone (iteratieve) oplossing die je vorig jaar hebt gemaakt (hier zijn Wt’s uitwerkingen overigens), vraag Wt eventueel om extra uitleg om de recursieve versie goed te snappen.

Heb je helder wat recursie is? Mooi, dan kun je aan de slag.